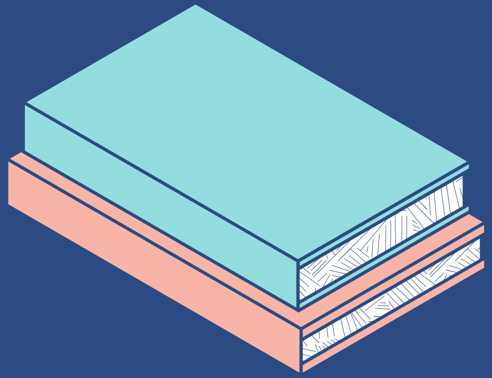
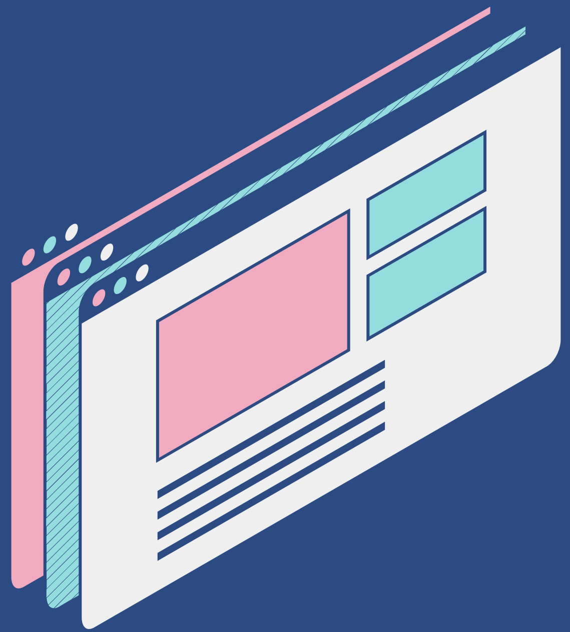
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY



Object-oriented programming

ELECTRICAL CIRCUIT SIMULATOR

By Group 31



Our team (Group 31):

Nguyen Quang Anh 20176684: Diagram designer.

Bui Huu Thanh Cong 20205176: Presenter.

Nguyen Hoang Minh 20226057: Team leader, coding, moderator.

Le Ngoc Viet 20205175: Report & slide designer.

TABLE OF CONTENTS

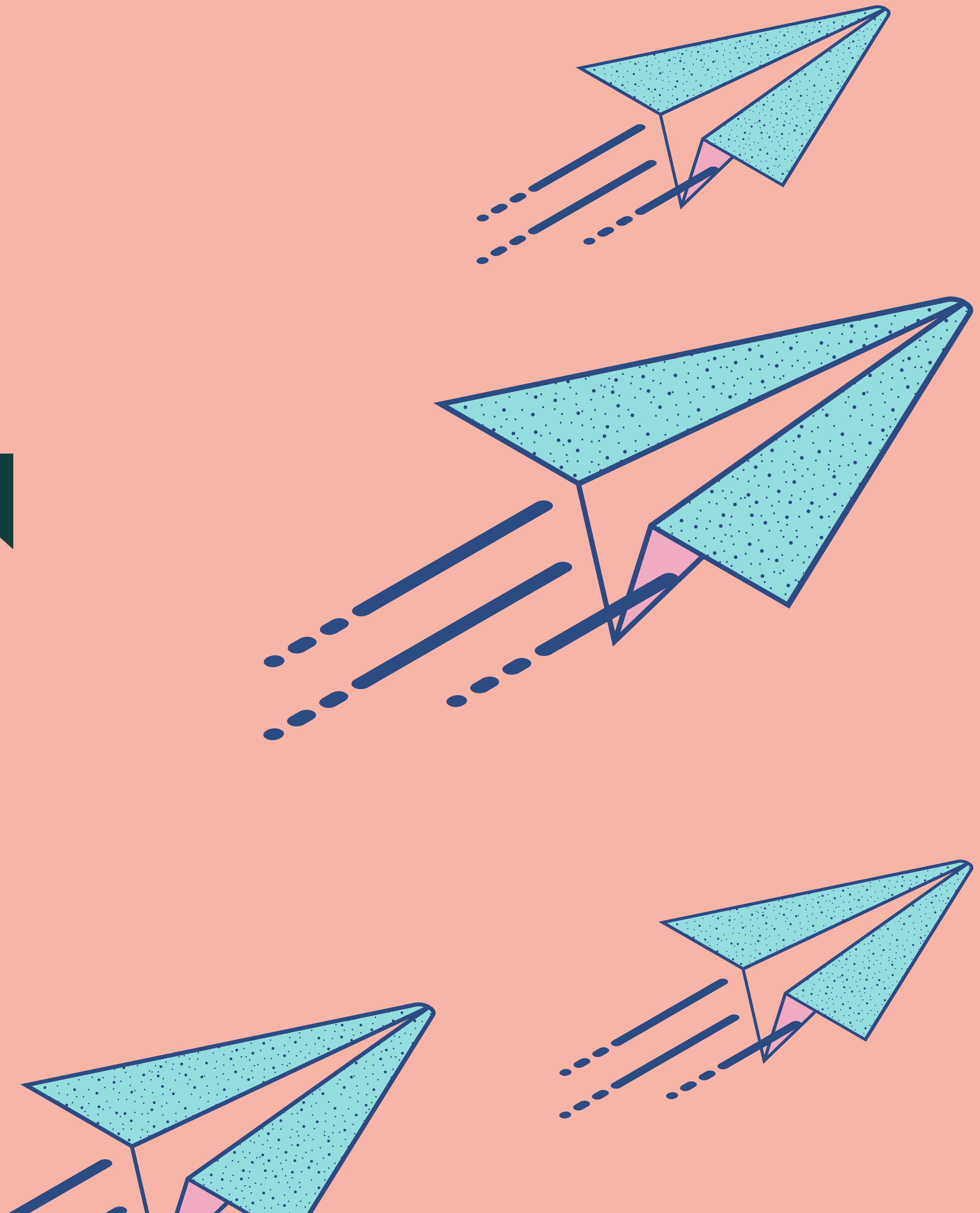
I. Introduction

II. Diagram

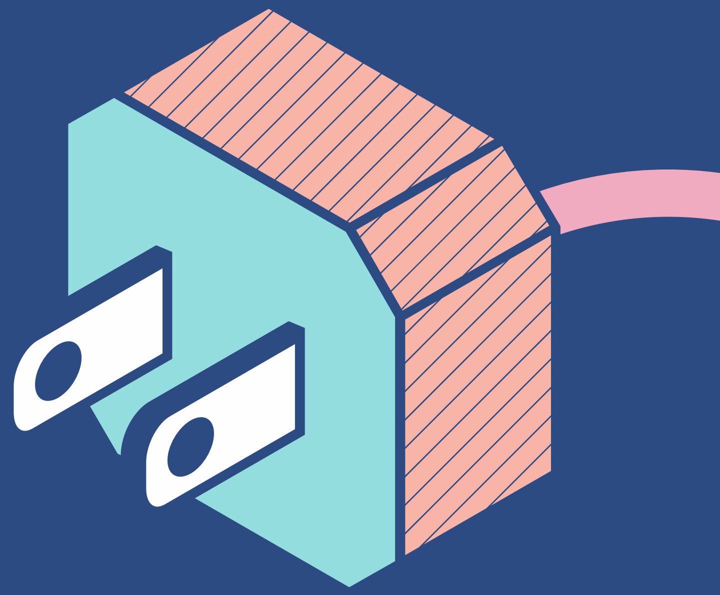
III. Implementation details

IV. OOP analysis

I. INTRODUCTION

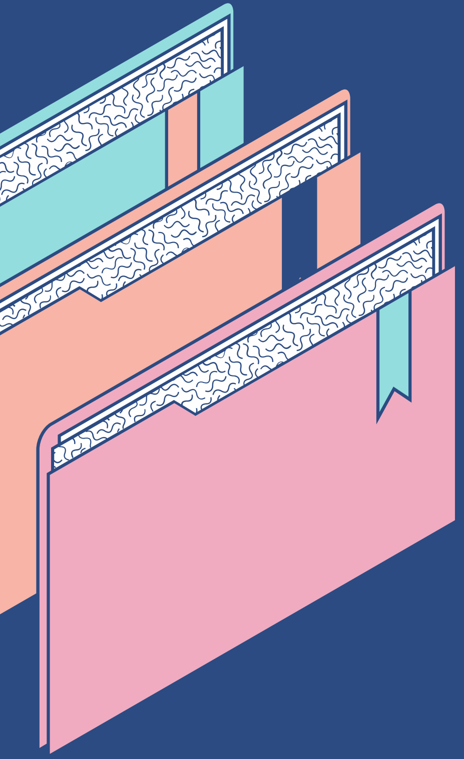


Project Overview



- This project develops an electrical circuit simulator that let users design and analyze circuits with resistors, capacitors, and inductors, using AC or DC sources.

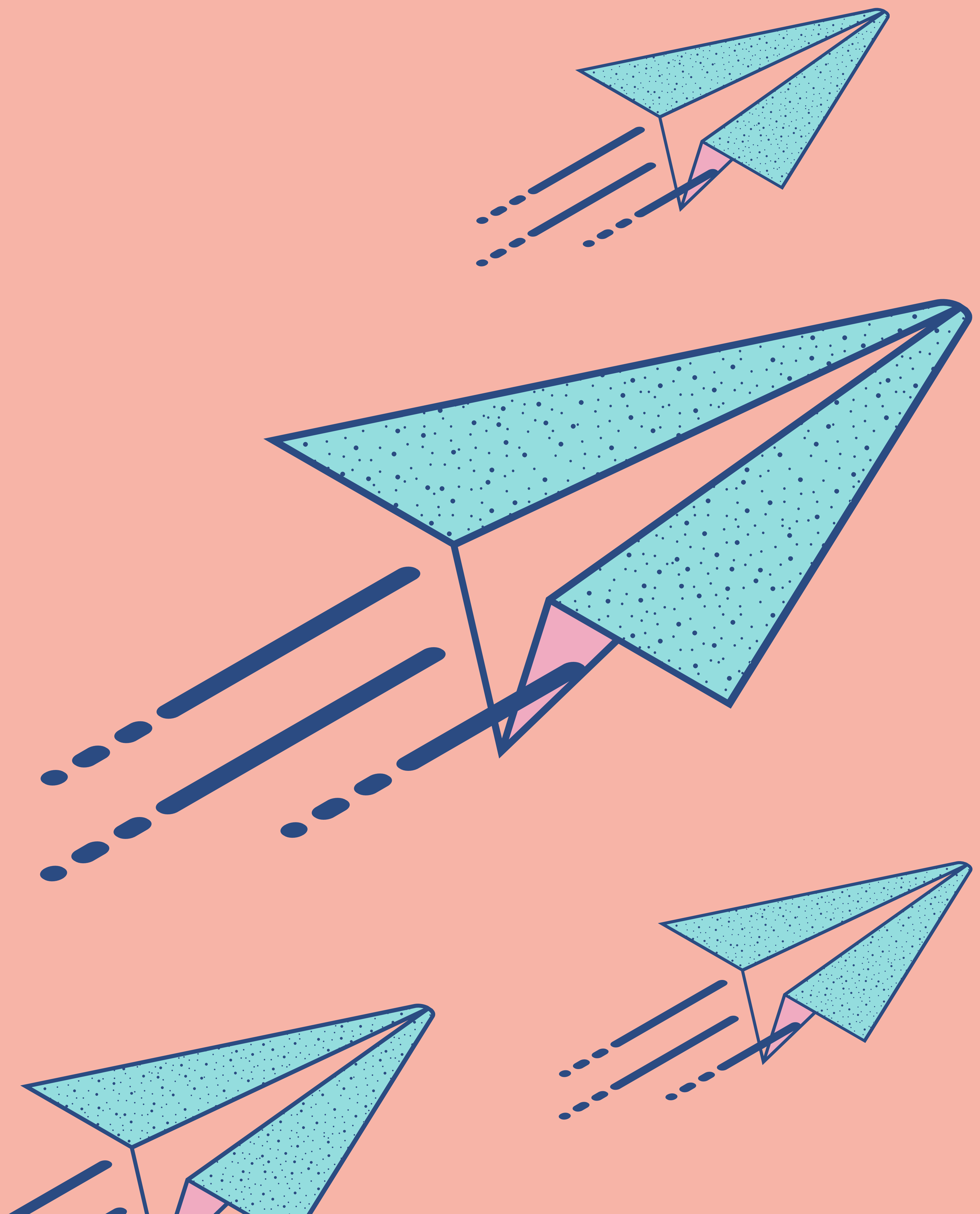
- It computes and displays voltage, current, and resistance for each component and provides visual circuit representations.



Objectives

- Provide a user-friendly interface for designing electrical circuits.
- Enable real-time calculations of circuit parameters.
- Support both serial and parallel circuit configurations.
- Visualize the designed circuits and display calculated values.
- Support users on the principles of circuit design and analysis.

II. DIAGRAMS



Use Case Diagram

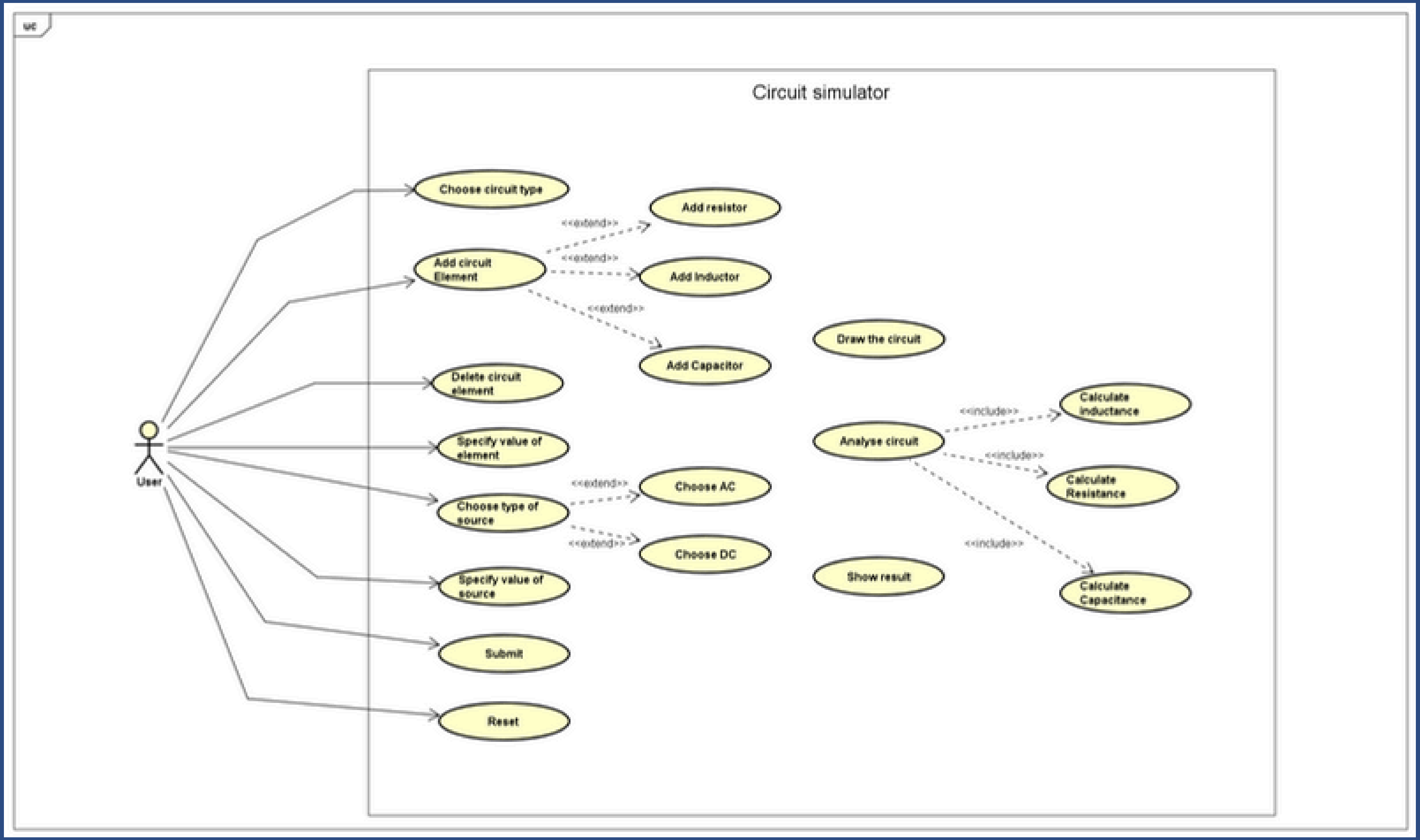
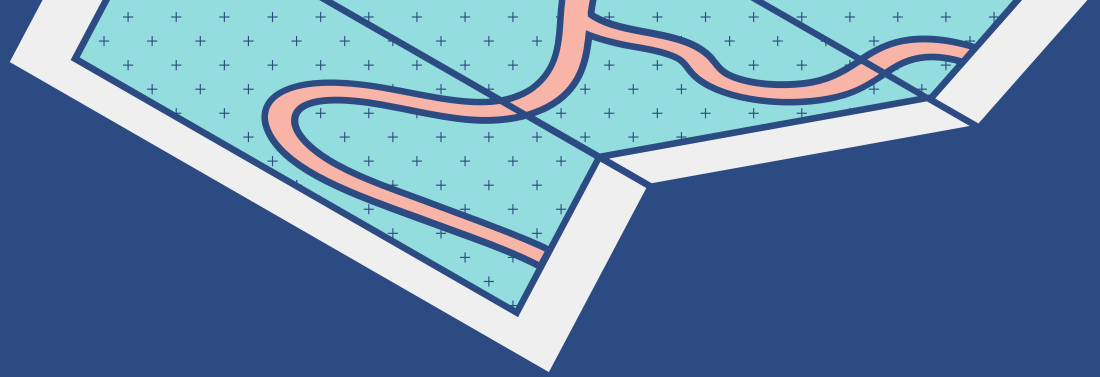


Figure 1: Use Case Diagram



Explanation:

Actor:

User is the actor.

Choose Circuit Type:

Users select the type of circuit they want to create (Serial or Parallel).

Add Circuit Element:

Users can add different elements to the circuit, such as resistors, inductors, and capacitors.

Delete Circuit Element:

Users can remove an element from the circuit.

Specify Value of Element:

Users specify the values for each added element.

Choose Type of Source:

Specify Value of Source: Reset the current circuit design.

Submit:

Submit the circuit design for analysis.

Reset:

Reset the current circuit design.

Class diagram

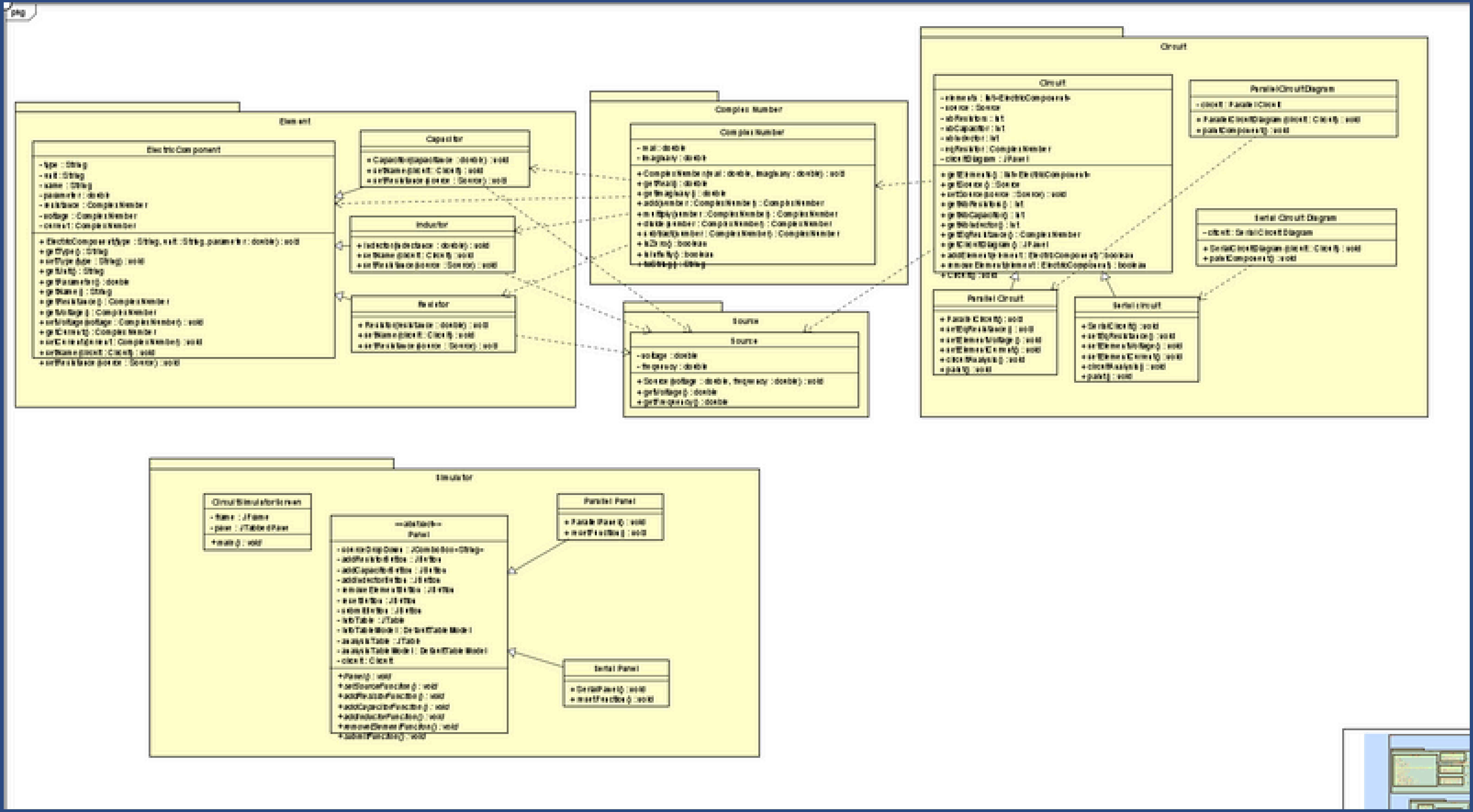
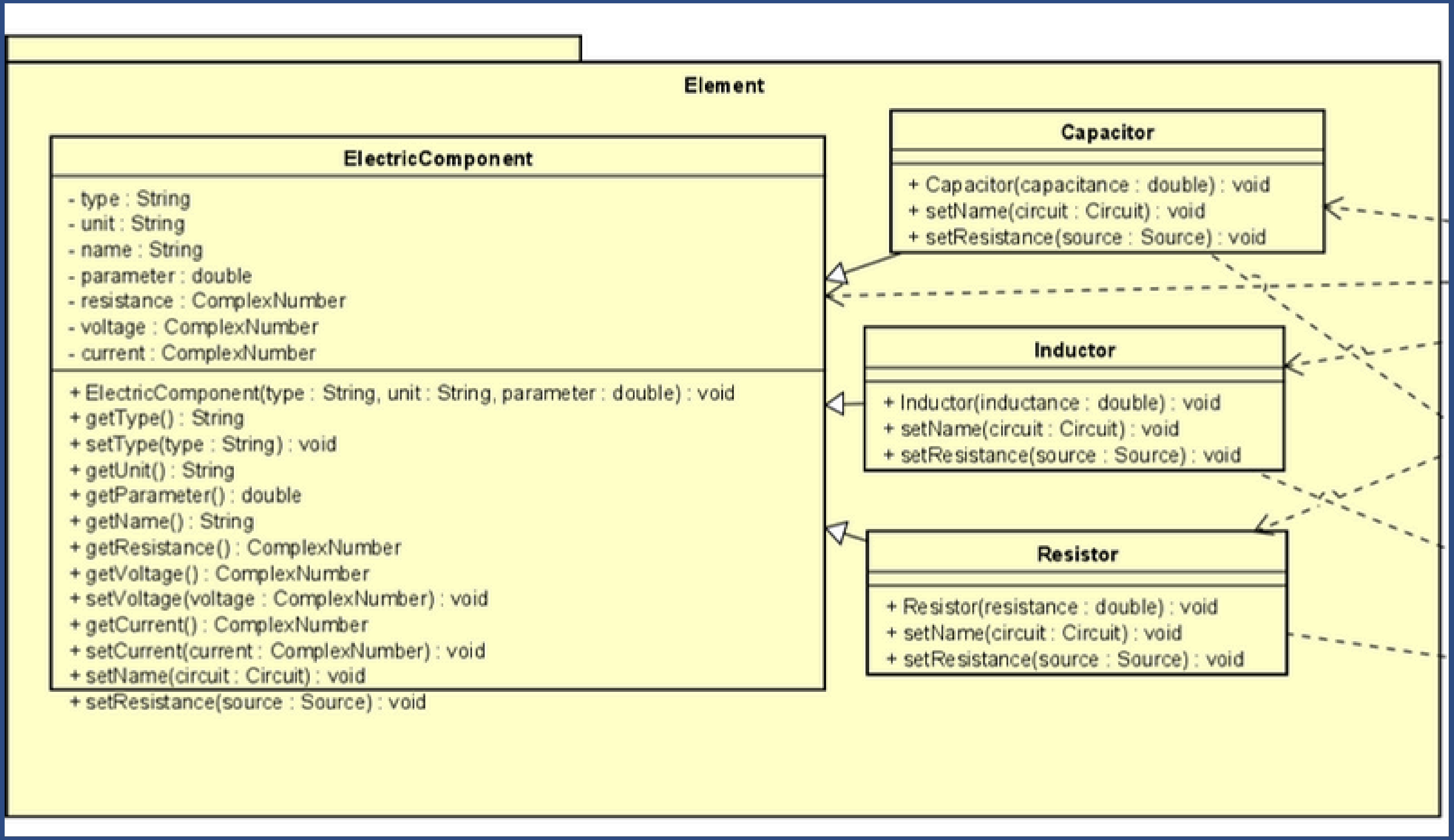

pkg
Element
ElectricComponent
- type : String
- unit : String
- name : String
- parameter : double
- resistance : ComplexNumber
- voltage : ComplexNumber
- current : ComplexNumber
+ ElectricComponent(type : String, unit : String, parameter : double) : void
+ getType() : String
+ setType(type : String) : void
+ getUnit() : String
+ getParameter() : double
+ getName() : String
+ getResistance() : ComplexNumber
+ getVoltage() : ComplexNumber
+ setVoltage(voltage : ComplexNumber) : void
+ getCurrent() : ComplexNumber
+ setCurrent(current : ComplexNumber) : void
+ setName(circuit : Circuit) : void
+ setResistance(source : Source) : void
Capacitor
+ Capacitor(capacitance : double) : void
+ setName(circuit : Circuit) : void
+ setResistance(source : Source) : void
Inductor
+ Inductor(inductance : double) : void
+ setName(circuit : Circuit) : void
+ setResistance(source : Source) : void
Resistor
+ Resistor(resistance : double) : void
+ setName(circuit : Circuit) : void
+ setResistance(source : Source) : void
Complex Number
ComplexNumber
- real : double
- imaginary : double
+ ComplexNumber(real : double, imaginary : double) : void
+ getReal() : double
+ getImaginary() : double
+ add(number : ComplexNumber) : ComplexNumber
+ multiply(number : ComplexNumber) : ComplexNumber
+ divide(number : ComplexNumber) : ComplexNumber
+ subtract(number : ComplexNumber) : ComplexNumber
+ isZero() : boolean
+ isInfinity() : boolean
+ toString() : String
Source
Source
- voltage : double
- frequency : double
+ Source(voltage : double, frequency : double) : void
+ getVoltage() : double
+ getFrequency() : double
Circuit
Circuit
- elements : List<ElectricComponent>
- source : Source
- nbResistors : int
- nbCapacitor : int
- nbInductor : int
- eqResistor : ComplexNumber
- circuitDiagram : JPanel
+ getElements() : List<ElectricComponent>
+ getSource() : Source
+ setSource(source : Source) : void
+ getNbResistors() : int
+ getNbCapacitor() : int
+ getNbInductor() : int
+ getEqResistance() : ComplexNumber
+ getCircuitDiagram() : JPanel
+ addElement(element : ElectricComponent) : boolean
+ removeElement(element : ElectricComponent) : boolean
+ Circuit() : void
ParallelCircuitDiagram
- circuit : ParallelCircuit
+ ParallelCircuitDiagram(circuit : Circuit) : void
+ paintComponent() : void
Serial Circuit Diagram
- circuit : SerialCircuitDiagram
+ SerialCircuitDiagram(circuit : Circuit) : void
+ paintComponent() : void
Parallel Circuit
+ ParallelCircuit() : void
+ setEqResistance() : void
+ setElementsVoltage() : void
+ setElementsCurrent() : void
+ circuitAnalysis() : void
+ paint() : void
Serial circuit
+ SerialCircuit() : void
+ setEqResistance() : void
+ setElementsVoltage() : void
+ setElementsCurrent() : void
+ circuitAnalysis() : void
+ paint() : void
Simulator
CircuitSimulatorScreen
- frame : JFrame
- pane : JTabbedPane
+ main() : void
<<abstract>>
Panel
- sourceDropDown : JComboBox<String>
- addResistorButton : JButton
- addCapacitorButton : JButton
- addInductorButton : JButton
- removeElementButton : JButton
- insertButton : JButton
- submitButton : JButton
- tabTable : JTable
- tabTableModel : DefaultTableModel
- analysisTable : JTable
- analysisTableModel : DefaultTableModel
- circuit : Circuit
+ Panel() : void
+ setSourceFunction() : void
+ addResistorFunction() : void
+ addCapacitorFunction() : void
+ addInductorFunction() : void
+ removeElementFunction() : void
+ submitFunction() : void
Parallel Panel
+ ParallelPanel() : void
+ insertFunction() : void
Serial Panel
+ SerialPanel() : void
+ insertFunction() : void

Figure 2: General Class Diagram

Package

ELEMENT PACKAGE

- Electric Component: Abstract class for generic electrical components with attributes like type, name, parameter, resistance, voltage, and current.
- Resistor, Capacitor, Inductor: Subclasses of ElectricComponent with methods to set values and calculate parameters.



Element

ElectricComponent

- type : String
- unit : String
- name : String
- parameter : double
- resistance : ComplexNumber
- voltage : ComplexNumber
- current : ComplexNumber

+ ElectricComponent(type : String, unit : String, parameter : double) : void
+ getType() : String
+ setType(type : String) : void
+ getUnit() : String
+ getParameter() : double
+ getName() : String
+ getResistance() : ComplexNumber
+ getVoltage() : ComplexNumber
+ setVoltage(voltage : ComplexNumber) : void
+ getCurrent() : ComplexNumber
+ setCurrent(current : ComplexNumber) : void
+ setName(circuit : Circuit) : void
+ setResistance(source : Source) : void

Capacitor

+ Capacitor(capacitance : double) : void
+ setName(circuit : Circuit) : void
+ setResistance(source : Source) : void

Inductor

+ Inductor(inductance : double) : void
+ setName(circuit : Circuit) : void
+ setResistance(source : Source) : void

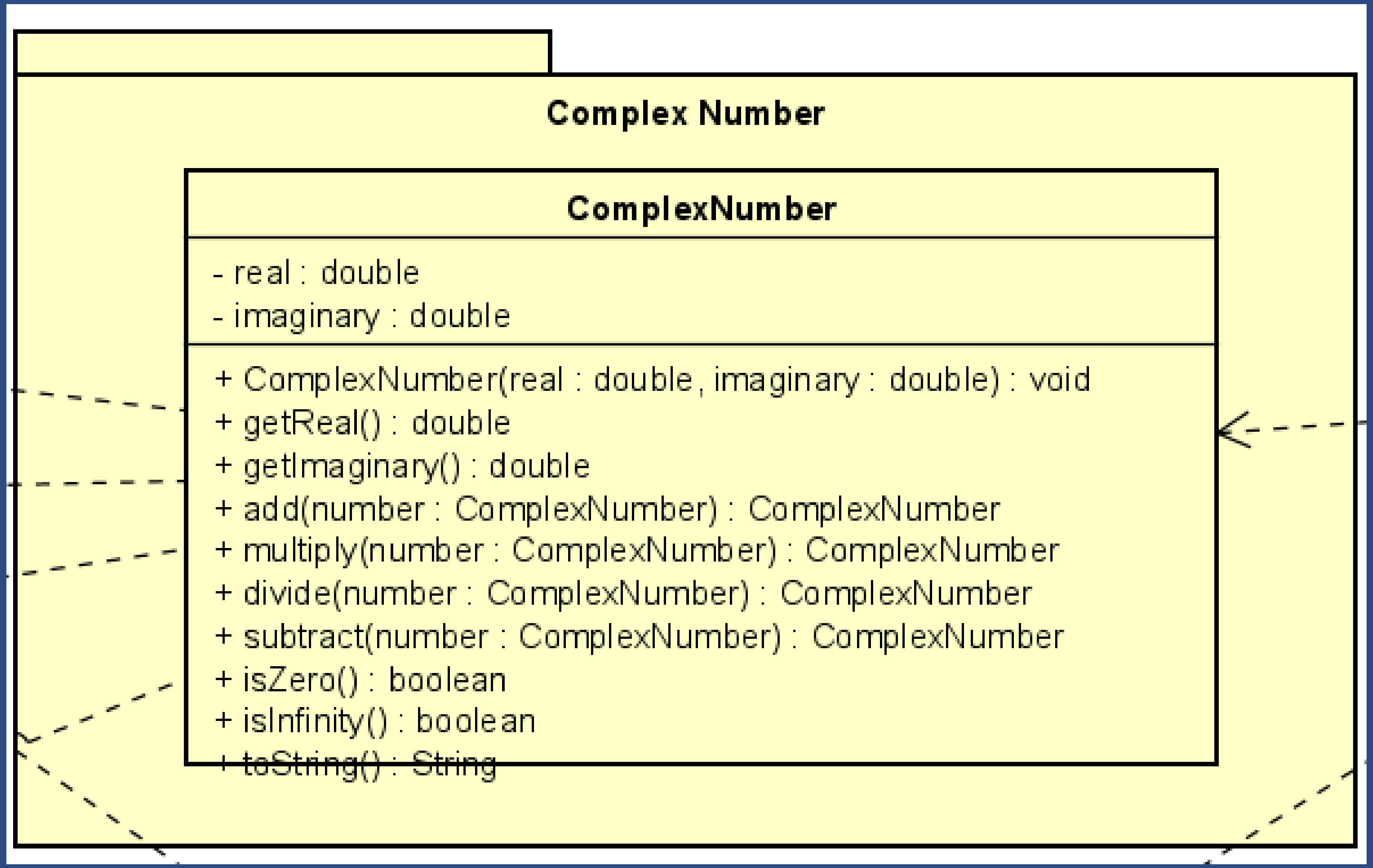
Resistor

+ Resistor(resistance : double) : void
+ setName(circuit : Circuit) : void
+ setResistance(source : Source) : void

Package

COMPLEX_NUMBER PACKAGE

- ComplexNumber: Class for handling complex number operations (addition, subtraction, multiplication, division) necessary for circuit calculations (U, I, R), including checks for zero or infinity, and string representation.



Complex Number

ComplexNumber

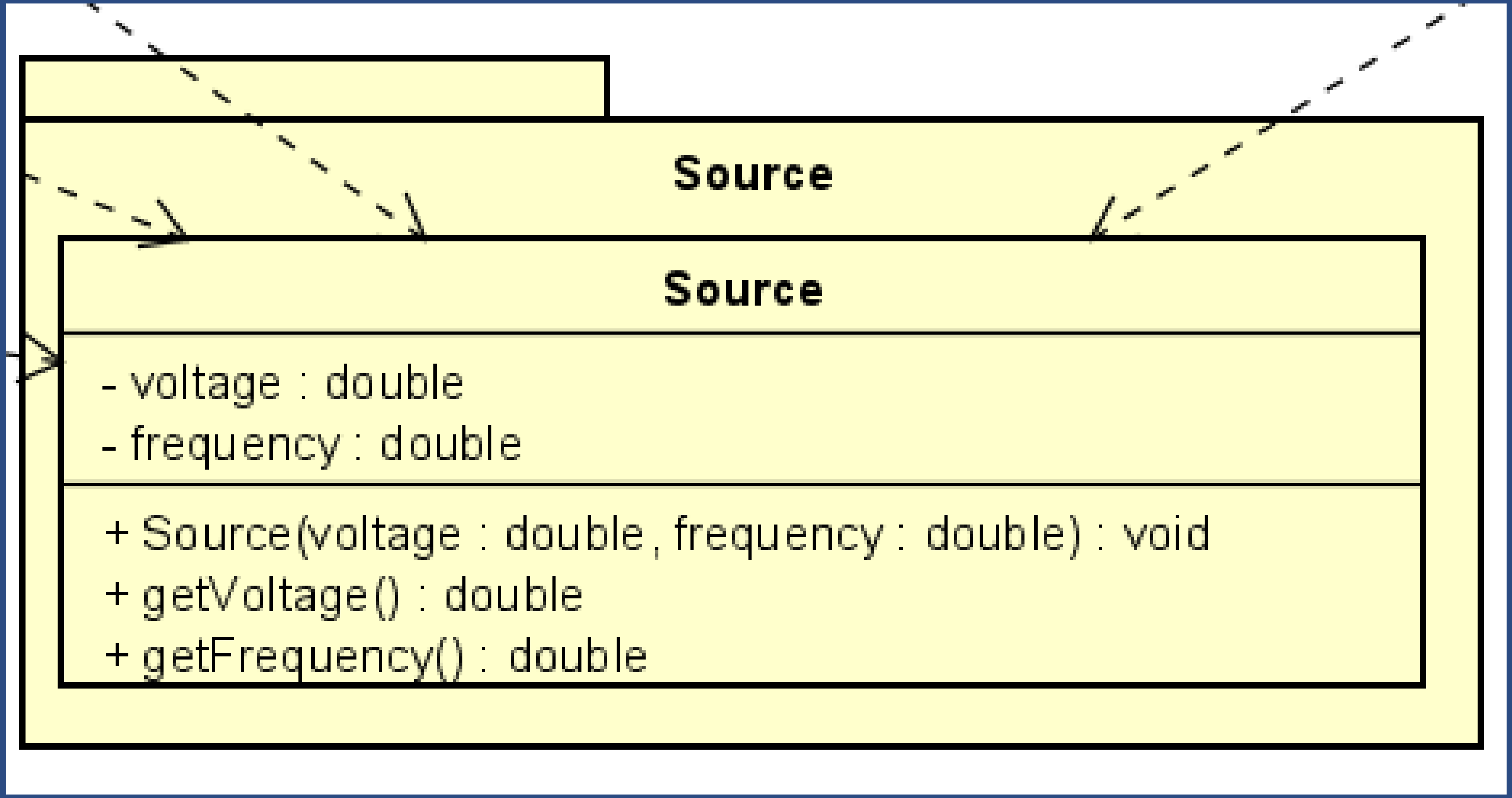
- real : double
- imaginary : double

+ ComplexNumber(real : double, imaginary : double) : void
+ getReal() : double
+ getImaginary() : double
+ add(number : ComplexNumber) : ComplexNumber
+ multiply(number : ComplexNumber) : ComplexNumber
+ divide(number : ComplexNumber) : ComplexNumber
+ subtract(number : ComplexNumber) : ComplexNumber
+ isZero() : boolean
+ isInfinity() : boolean
+ toString() : String

Package

VOLTAGE_SOURCE PACKAGE

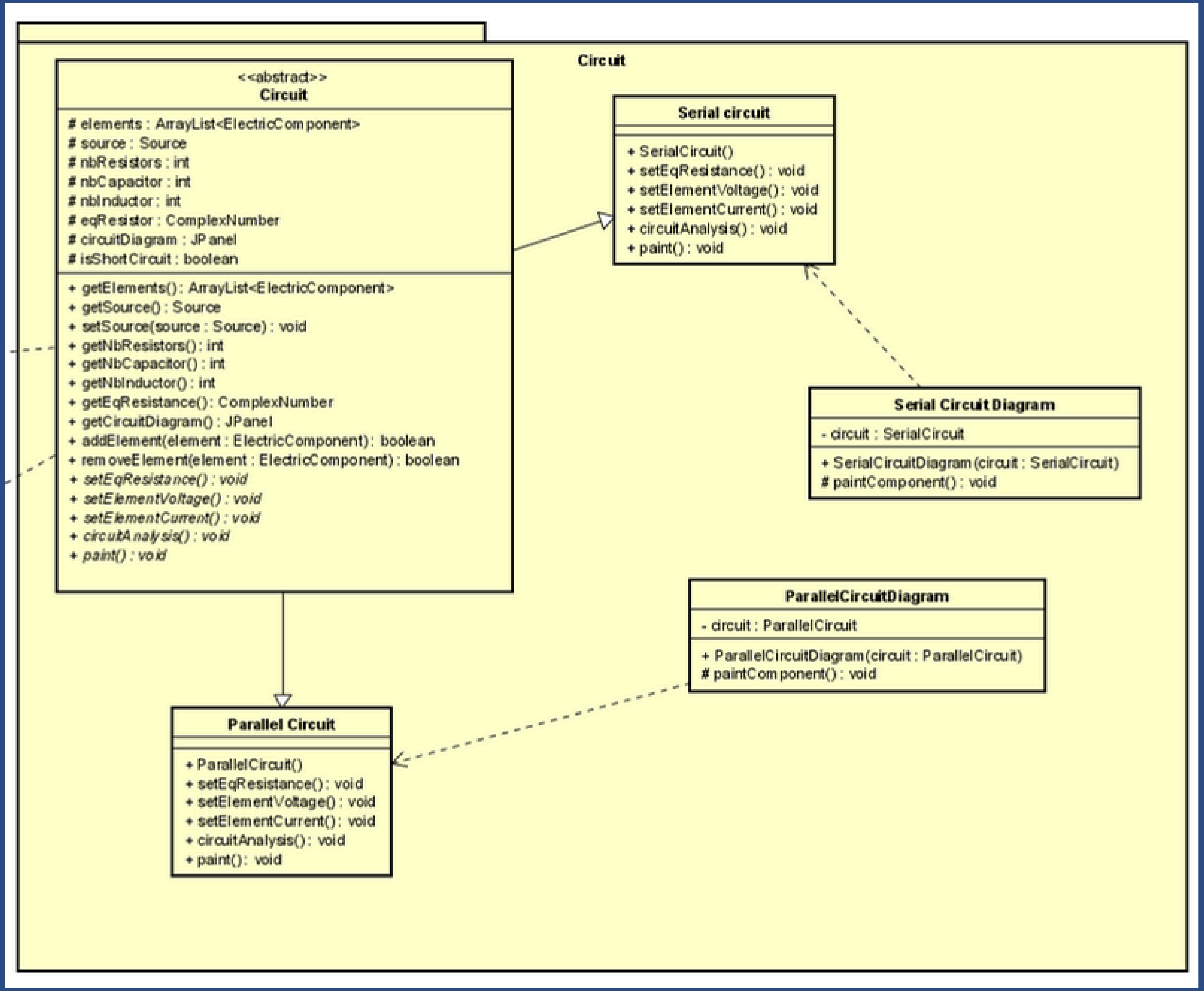
- Source: Class representing the circuit's power source, with attributes for voltage and frequency, and getter/setter methods.


Source
Source
- voltage : double
- frequency : double
+ Source(voltage : double, frequency : double) : void
+ getVoltage() : double
+ getFrequency() : double

Package

CIRCUIT PACKAGE

- Circuit: Abstract class for managing circuit components and power source, with methods to add/remove components and calculate overall circuit parameters.
- SerialCircuit & ParallelCircuit: Subclasses handling specific calculations for serial and parallel circuits.
- SerialCircuitDiagram & ParallelCircuitDiagram: Classes responsible for drawing the respective circuit types.



Circuit

<<abstract>>
Circuit

elements : ArrayList<ElectricComponent>
source : Source
nbResistors : int
nbCapacitor : int
nbInductor : int
eqResistor : ComplexNumber
circuitDiagram : JPanel
isShortCircuit : boolean

+ getElements(): ArrayList<ElectricComponent>
+ getSource() : Source
+ setSource(source : Source) : void
+ getNbResistors() : int
+ getNbCapacitor() : int
+ getNbInductor() : int
+ getEqResistance(): ComplexNumber
+ getCircuitDiagram() : JPanel
+ addElement(element : ElectricComponent): boolean
+ removeElement(element : ElectricComponent) : boolean
+ *setEqResistance() : void*
+ *setElementVoltage() : void*
+ *setElementCurrent() : void*
+ *circuitAnalysis() : void*
+ *paint() : void*

Serial circuit

+ SerialCircuit()
+ setEqResistance() : void
+ setElementVoltage() : void
+ setElementCurrent() : void
+ circuitAnalysis() : void
+ paint() : void

Serial Circuit Diagram

- circuit : SerialCircuit

+ SerialCircuitDiagram(circuit : SerialCircuit)
# paintComponent() : void

ParallelCircuitDiagram

- circuit : ParallelCircuit

+ ParallelCircuitDiagram(circuit : ParallelCircuit)
# paintComponent() : void

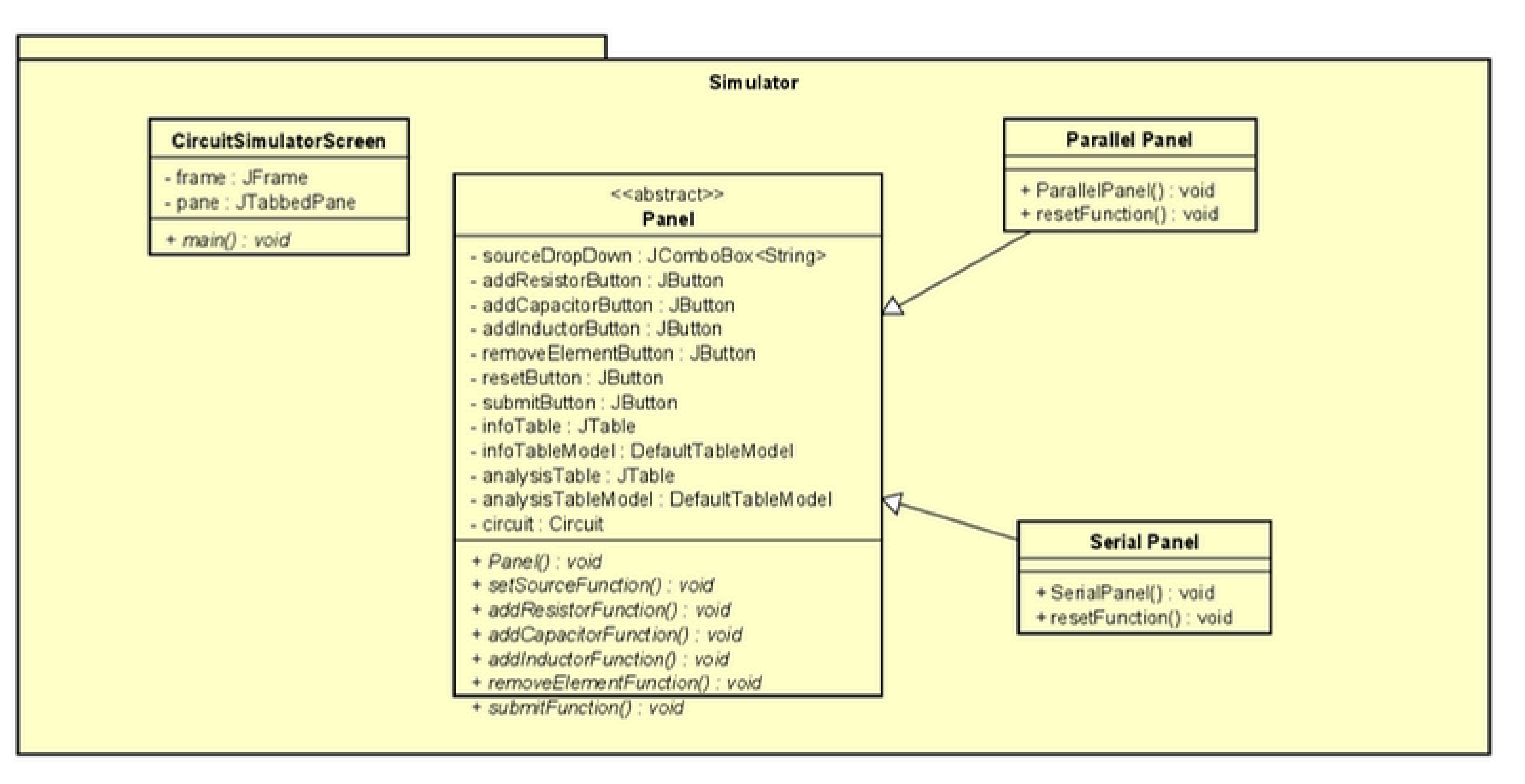
Parallel Circuit

+ ParallelCircuit()
+ setEqResistance(): void
+ setElementVoltage() : void
+ setElementCurrent(): void
+ circuitAnalysis(): void
+ paint(): void

Package

SIMULATOR PACKAGE

- CircuitSimulatorScreen: Class for the main interface, allowing user interaction with the circuit, including adding/removing components and setting values.
- Panel: Abstract class for interface panels.
- ParallelPanel, SerialPanel: Subclasses managing user inputs for parallel and serial circuits, respectively.



Simulator

CircuitSimulatorScreen

- frame : JFrame
- pane : JTabbedPane

+ *main() : void*

<<abstract>>
Panel

- sourceDropDown : JComboBox<String>
- addResistorButton : JButton
- addCapacitorButton : JButton
- addInductorButton : JButton
- removeElementButton : JButton
- resetButton : JButton
- submitButton : JButton
- infoTable : JTable
- infoTableModel : DefaultTableModel
- analysisTable : JTable
- analysisTableModel : DefaultTableModel
- circuit : Circuit

+ *Panel() : void*
+ *setSourceFunction() : void*
+ *addResistorFunction() : void*
+ *addCapacitorFunction() : void*
+ *addInductorFunction() : void*
+ *removeElementFunction() : void*
+ *submitFunction() : void*

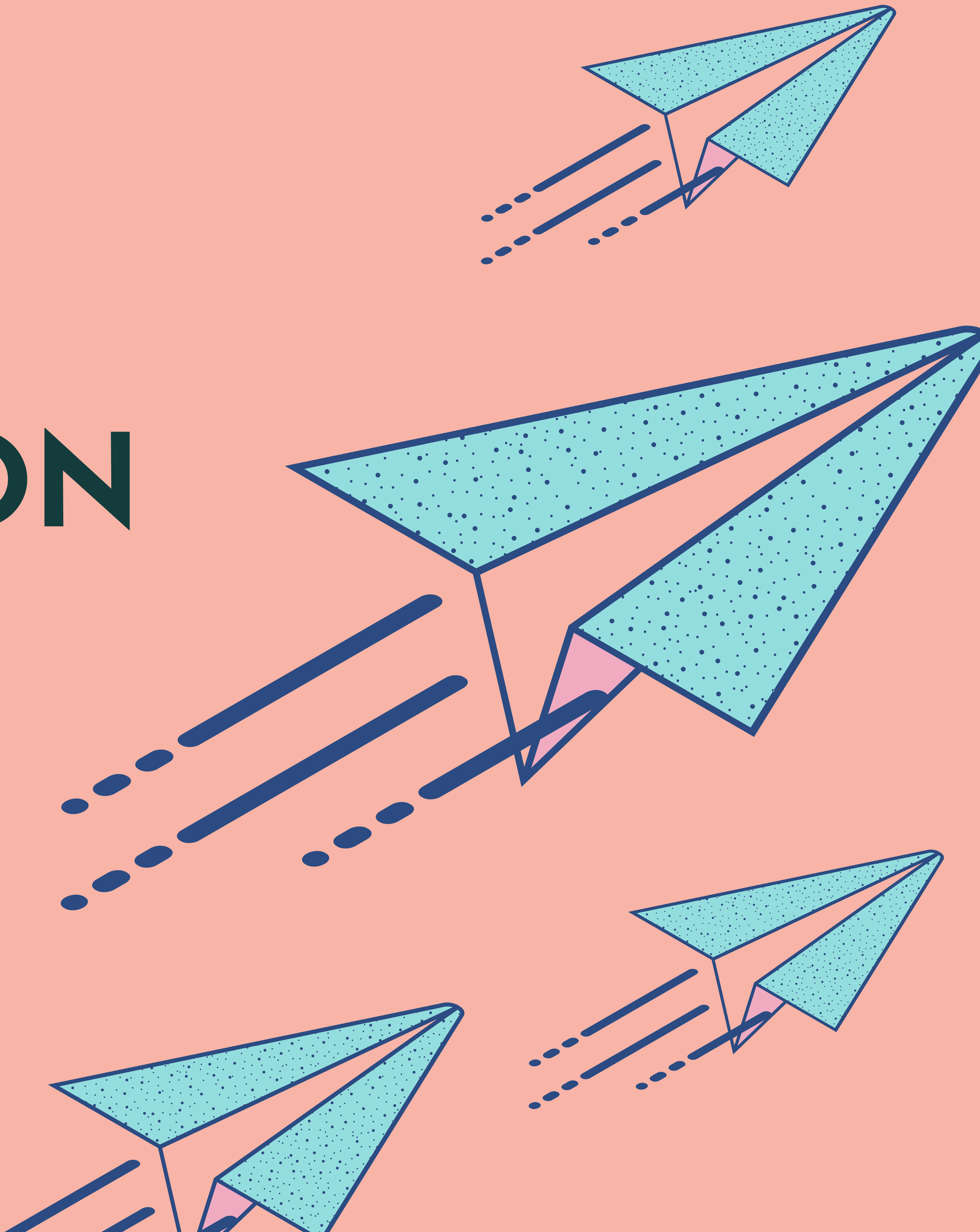
Parallel Panel

+ ParallelPanel() : void
+ resetFunction() : void

Serial Panel

+ SerialPanel() : void
+ resetFunction() : void

III. IMPLEMENTATION DETAILS



Core functionality

The core functionality of the application focuses on user interaction for creating and managing circuit designs. Key features include:

- **Input Validation**: Ensures users provide complete and correct information before simulation.
- **Time Calculations**: Computes voltage, current, and resistance as users modify the circuit.
- **Visualization**: Displays a graphical representation of the circuit.

User interface (UI)

The user interface (UI) allows users to add, modify, and submit circuit components effortlessly. Key UI components include:

Circuit Diagram:

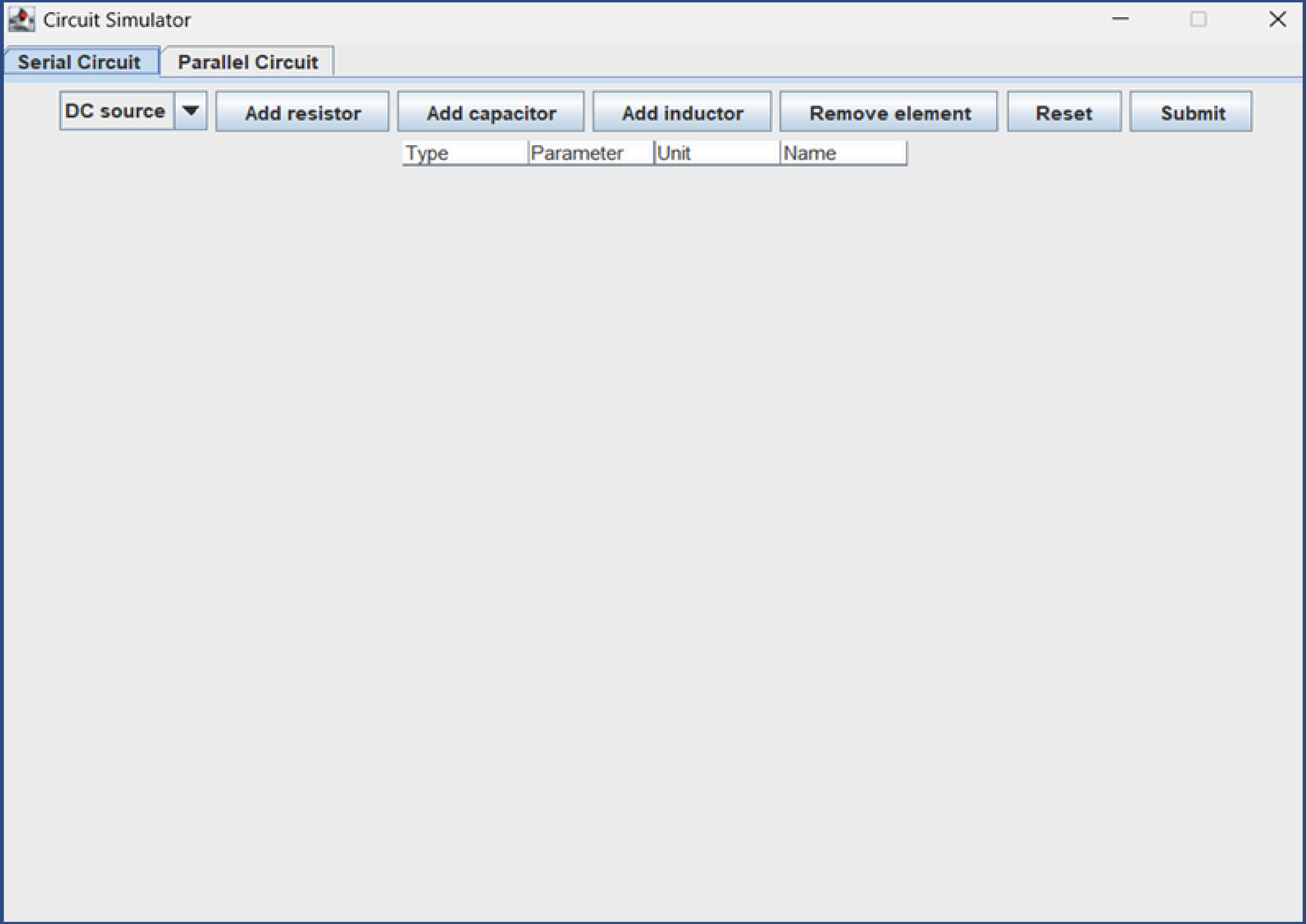
- Shows the layout of the circuit.

Component List:

- Displays all added components and their properties.

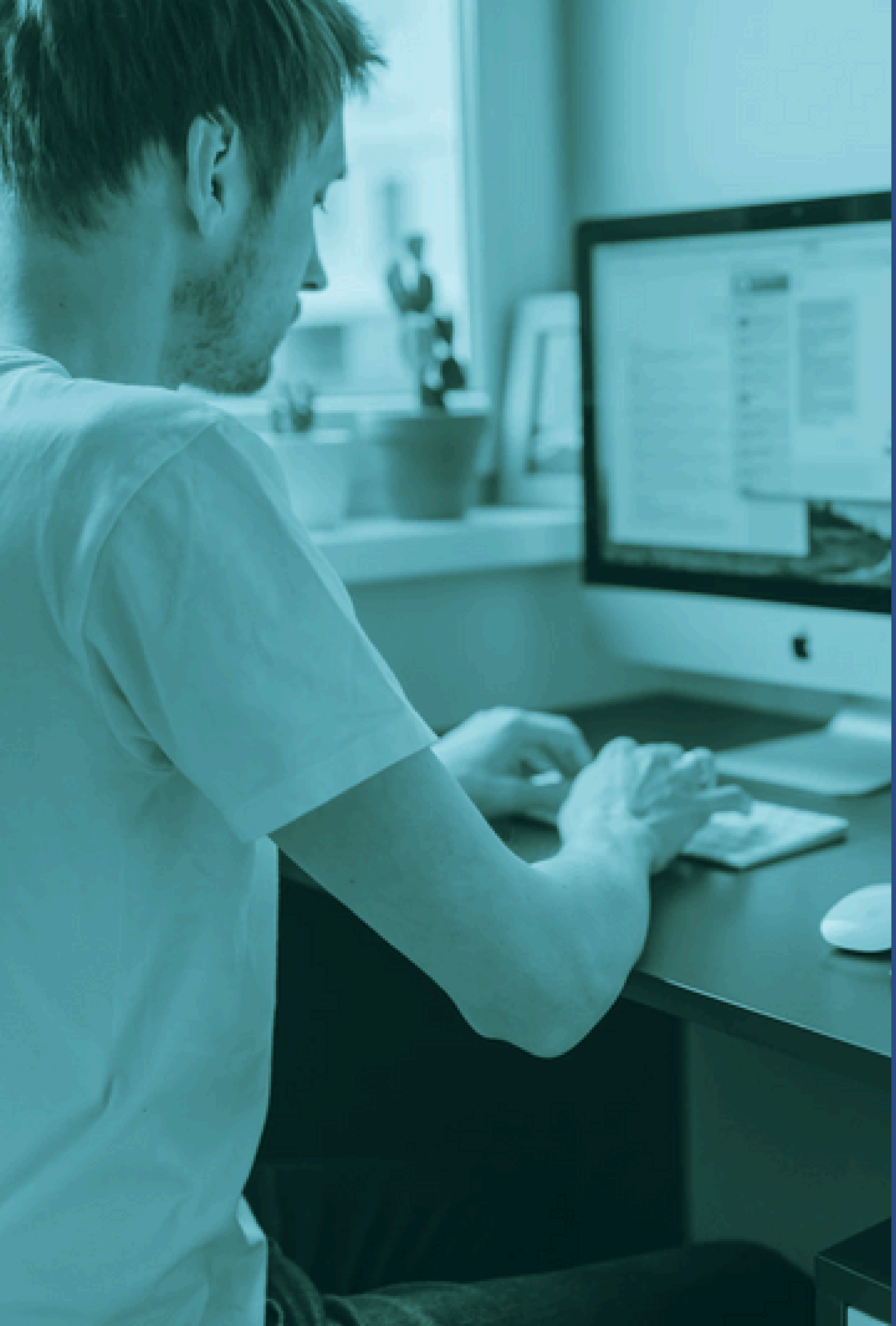
Control Buttons:

- Allows users to manage the circuit design process.



Type	Parameter	Unit	Name

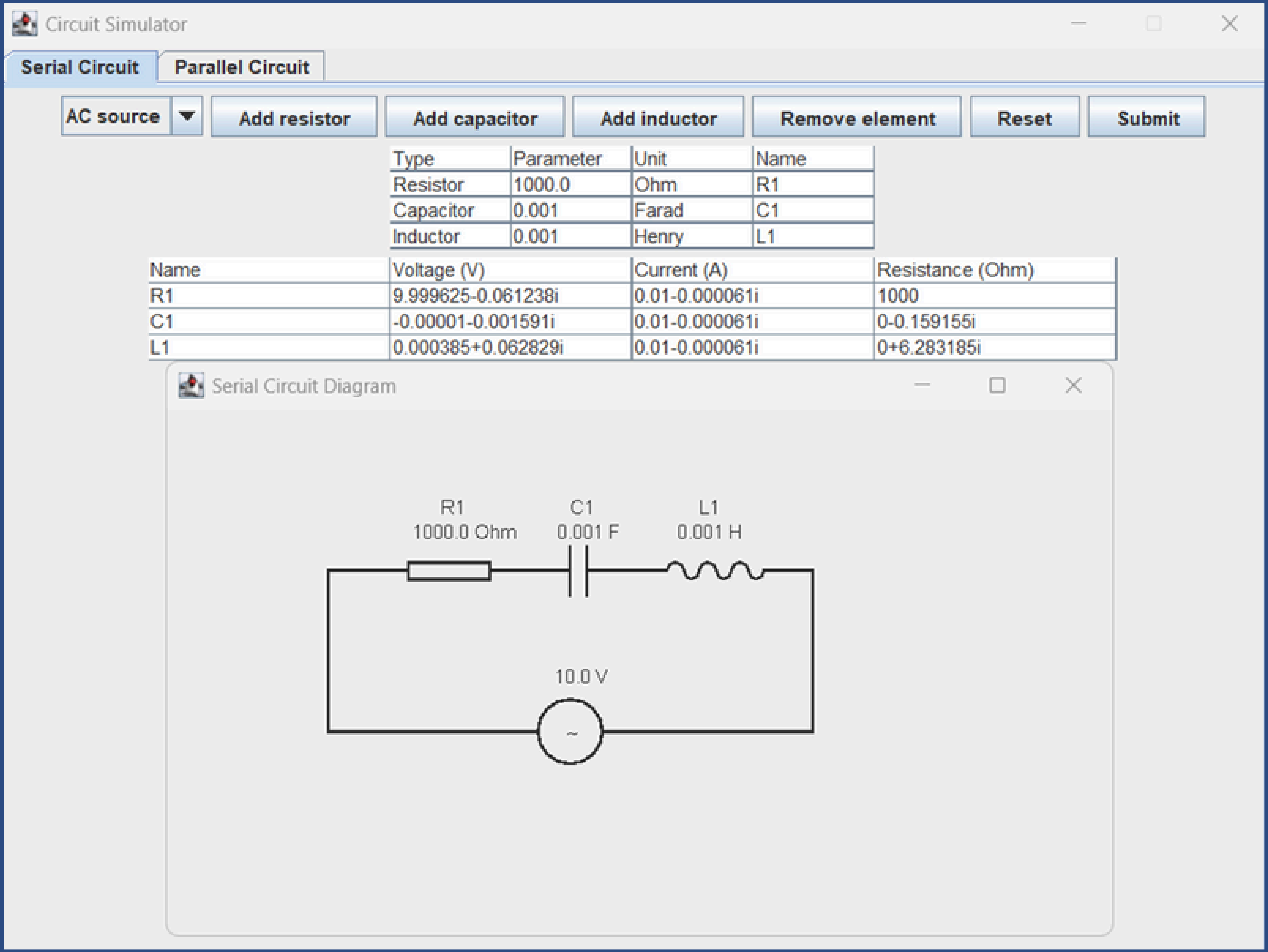
Figure 3: Program's main window



Calculation and analysis

The application performs real-time calculations for user-designed circuits, supporting both AC and DC circuits. Key calculation methods include:

- **Impedance Calculation**: Computes impedance for resistors, inductors, and capacitors.
- **Voltage and Current Calculation**: Uses Ohm's Law to determine values.
- **Short Circuit Detection**: Identifies and alerts users to potential short circuits.

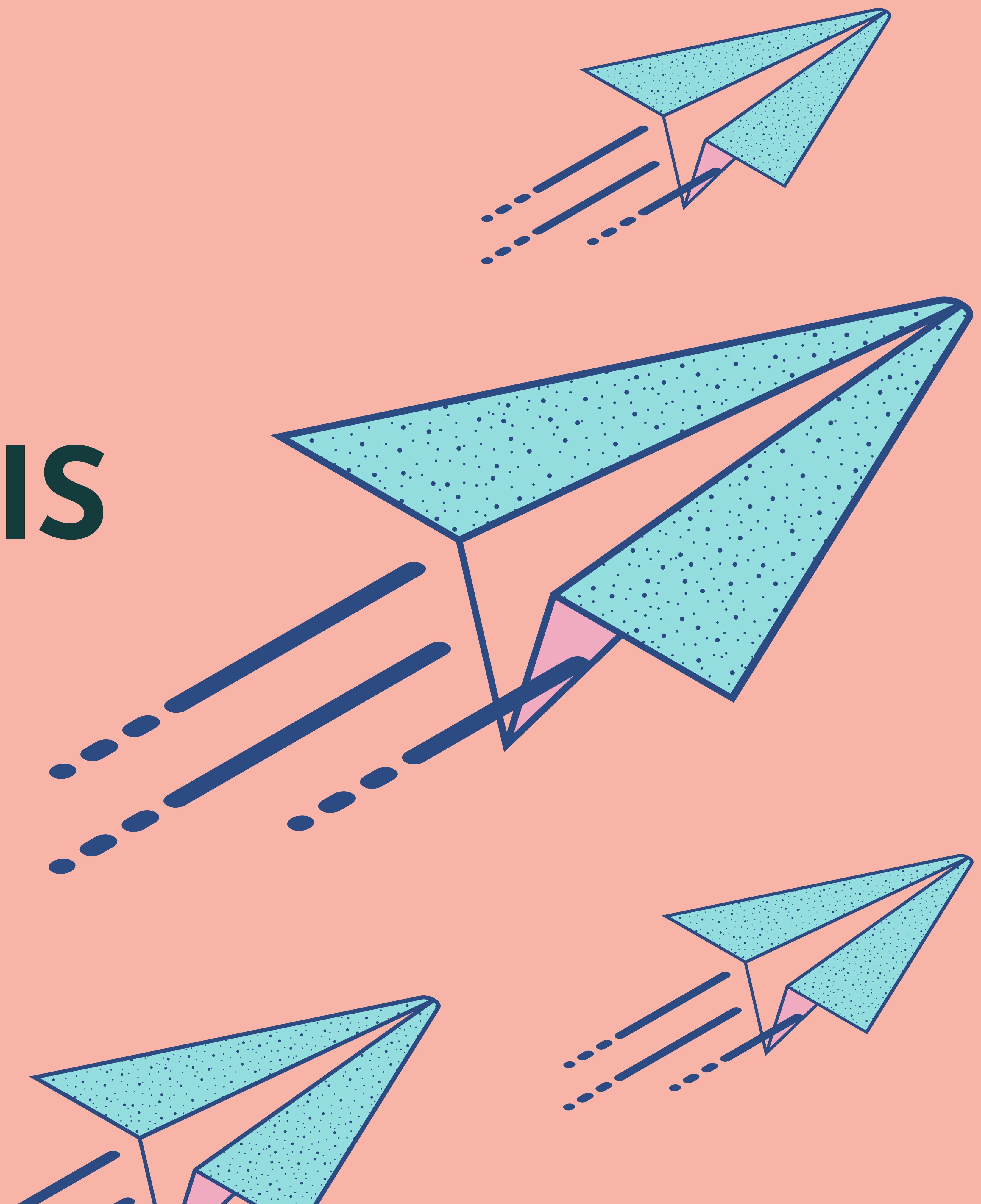


Type	Parameter	Unit	Name
Resistor	1000.0	Ohm	R1
Capacitor	0.001	Farad	C1
Inductor	0.001	Henry	L1

Name	Voltage (V)	Current (A)	Resistance (Ohm)
R1	9.999625-0.061238i	0.01-0.000061i	1000
C1	-0.00001-0.001591i	0.01-0.000061i	0-0.159155i
L1	0.000385+0.062829i	0.01-0.000061i	0+6.283185i

Figure 4: Calculation results.

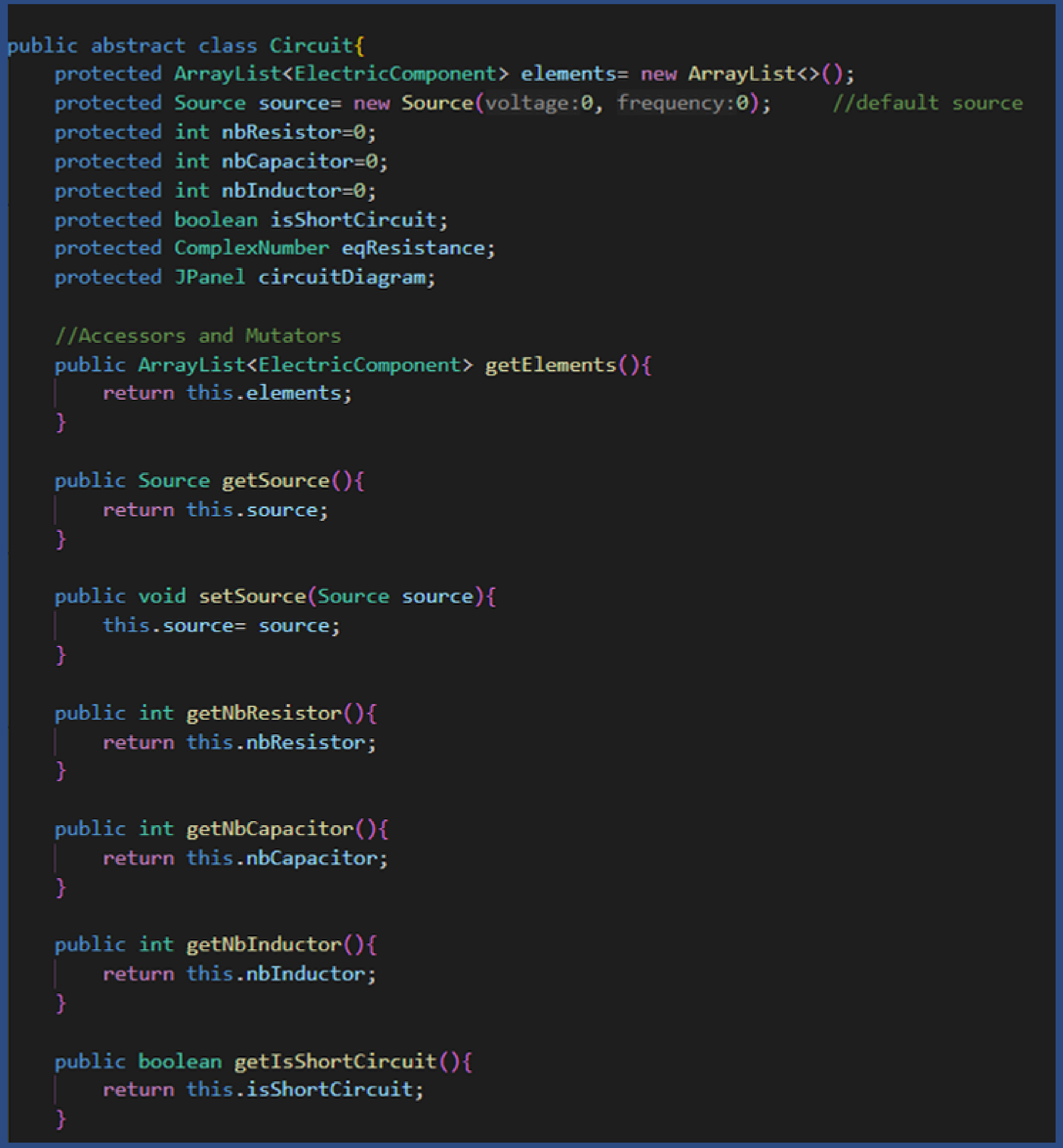
IV. OOP ANALYSIS



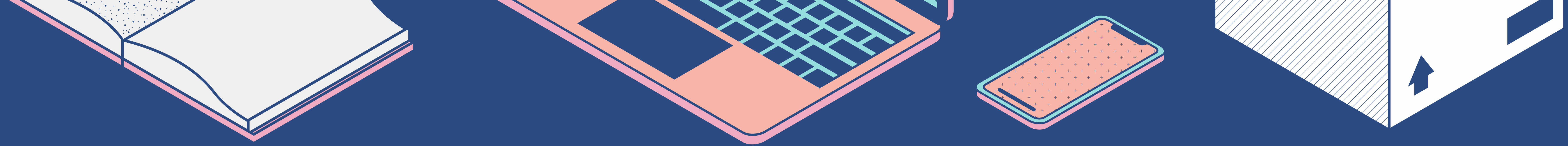
ENCAPSULATION

We hide the implementation details of a class by:

1. Declaring the attributes of a class as private/protected(for parent classes).
2. Defining getters and setters methods.
3. Packing the same-purpose classes in a package.



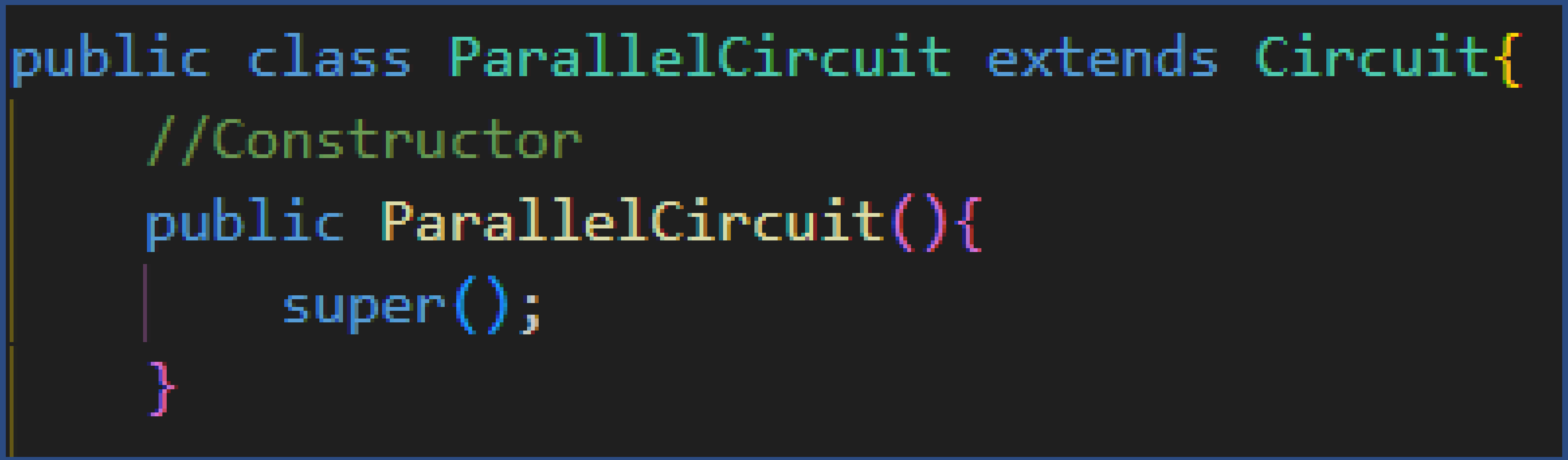
```java
public abstract class Circuit{
    protected ArrayList<ElectricComponent> elements= new ArrayList<>();
    protected Source source= new Source(voltage:0, frequency:0);     //default source
    protected int nbResistor=0;
    protected int nbCapacitor=0;
    protected int nbInductor=0;
    protected boolean isShortCircuit;
    protected ComplexNumber eqResistance;
    protected JPanel circuitDiagram;

    //Accessors and Mutators
    public ArrayList<ElectricComponent> getElements(){
        return this.elements;
    }

    public Source getSource(){
        return this.source;
    }

    public void setSource(Source source){
        this.source= source;
    }

    public int getNbResistor(){
        return this.nbResistor;
    }

    public int getNbCapacitor(){
        return this.nbCapacitor;
    }

    public int getNbInductor(){
        return this.nbInductor;
    }

    public boolean getIsShortCircuit(){
        return this.isShortCircuit;
    }
```



INHERITANCE

We use mono inheritance most in this project:

1. Using “extend” keyword.
2. Inheriting the parent’s methods and attributes.

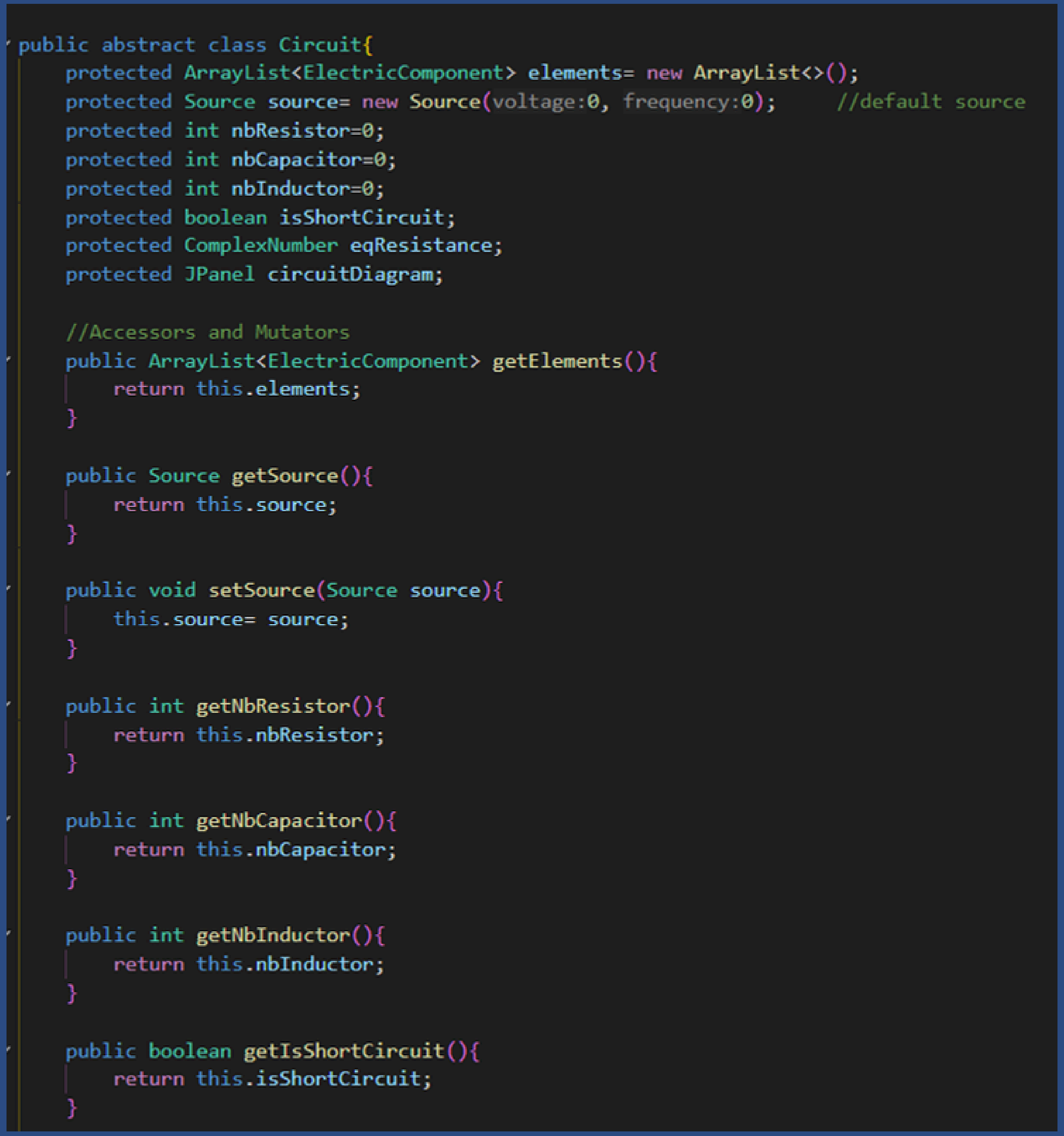


```
public class ParallelCircuit extends Circuit{
    //Constructor
    public ParallelCircuit(){
        super();
    }
```

ABSTRACTION

We use abstract classes in this project:

1. Circuit class.
2. ElectricComponent class.
3. Panel class.

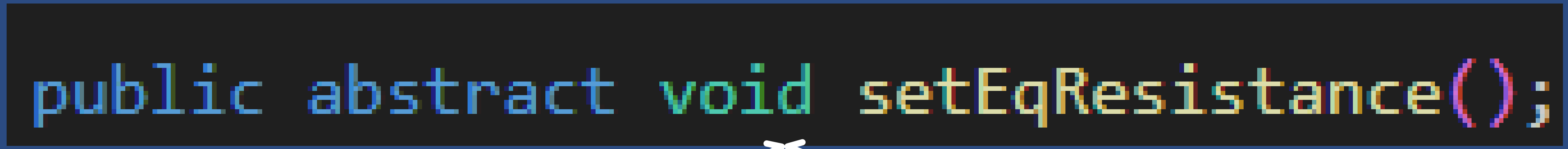


```
public abstract class Circuit{
    protected ArrayList<ElectricComponent> elements= new ArrayList<>();
    protected Source source= new Source(voltage:0, frequency:0);     //default source
    protected int nbResistor=0;
    protected int nbCapacitor=0;
    protected int nbInductor=0;
    protected boolean isShortCircuit;
    protected ComplexNumber eqResistance;
    protected JPanel circuitDiagram;

    //Accessors and Mutators
    public ArrayList<ElectricComponent> getElements(){
        return this.elements;
    }

    public Source getSource(){
        return this.source;
    }

    public void setSource(Source source){
        this.source= source;
    }

    public int getNbResistor(){
        return this.nbResistor;
    }

    public int getNbCapacitor(){
        return this.nbCapacitor;
    }

    public int getNbInductor(){
        return this.nbInductor;
    }

    public boolean getIsShortCircuit(){
        return this.isShortCircuit;
    }
```

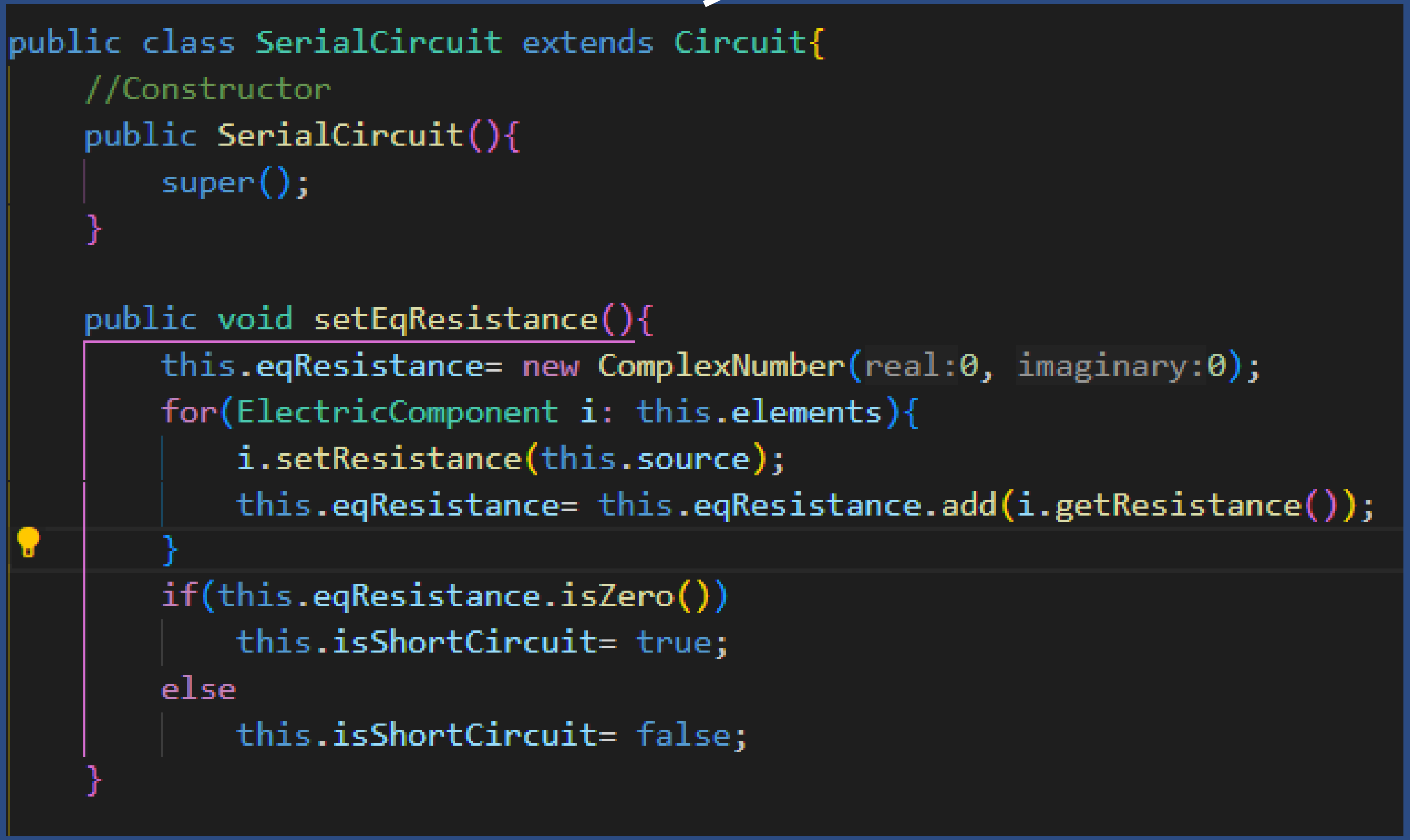
POLYMOPHISM

Subclass overrides the method of its parent:

- For example, in circuit package, ParallelCircuit and SerialCircuit overrides SetEqResistance method in Circuit class.
- This is because each type of circuit has different formula to achieve the equivalent resistance.

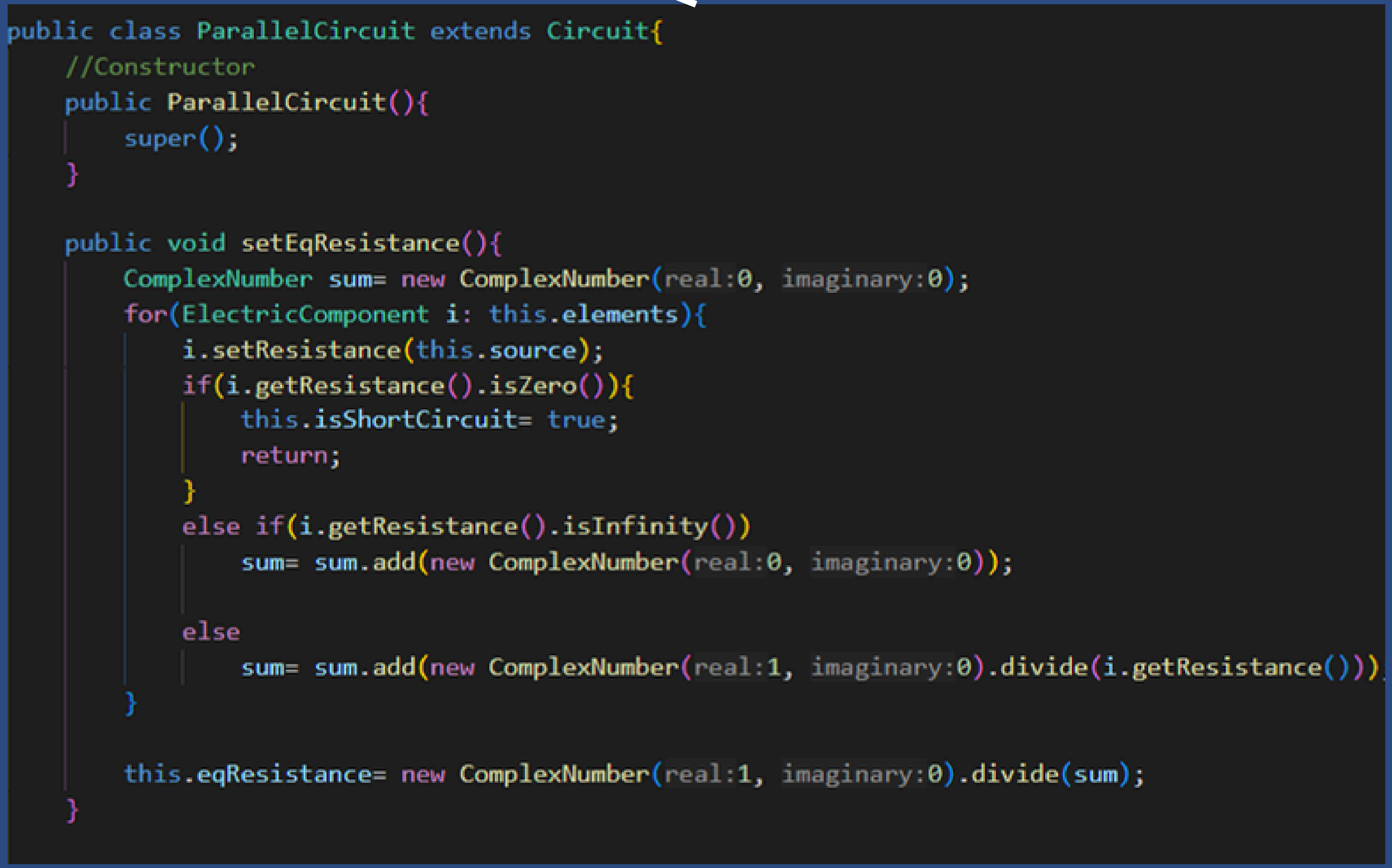


```java
public abstract void setEqResistance();
```



```java
public class SerialCircuit extends Circuit{
    //Constructor
    public SerialCircuit(){
        super();
    }

    public void setEqResistance(){
        this.eqResistance= new ComplexNumber(real:0, imaginary:0);
        for(ElectricComponent i: this.elements){
            i.setResistance(this.source);
            this.eqResistance= this.eqResistance.add(i.getResistance());
        }
        if(this.eqResistance.isZero())
            this.isShortCircuit= true;
        else
            this.isShortCircuit= false;
    }
```

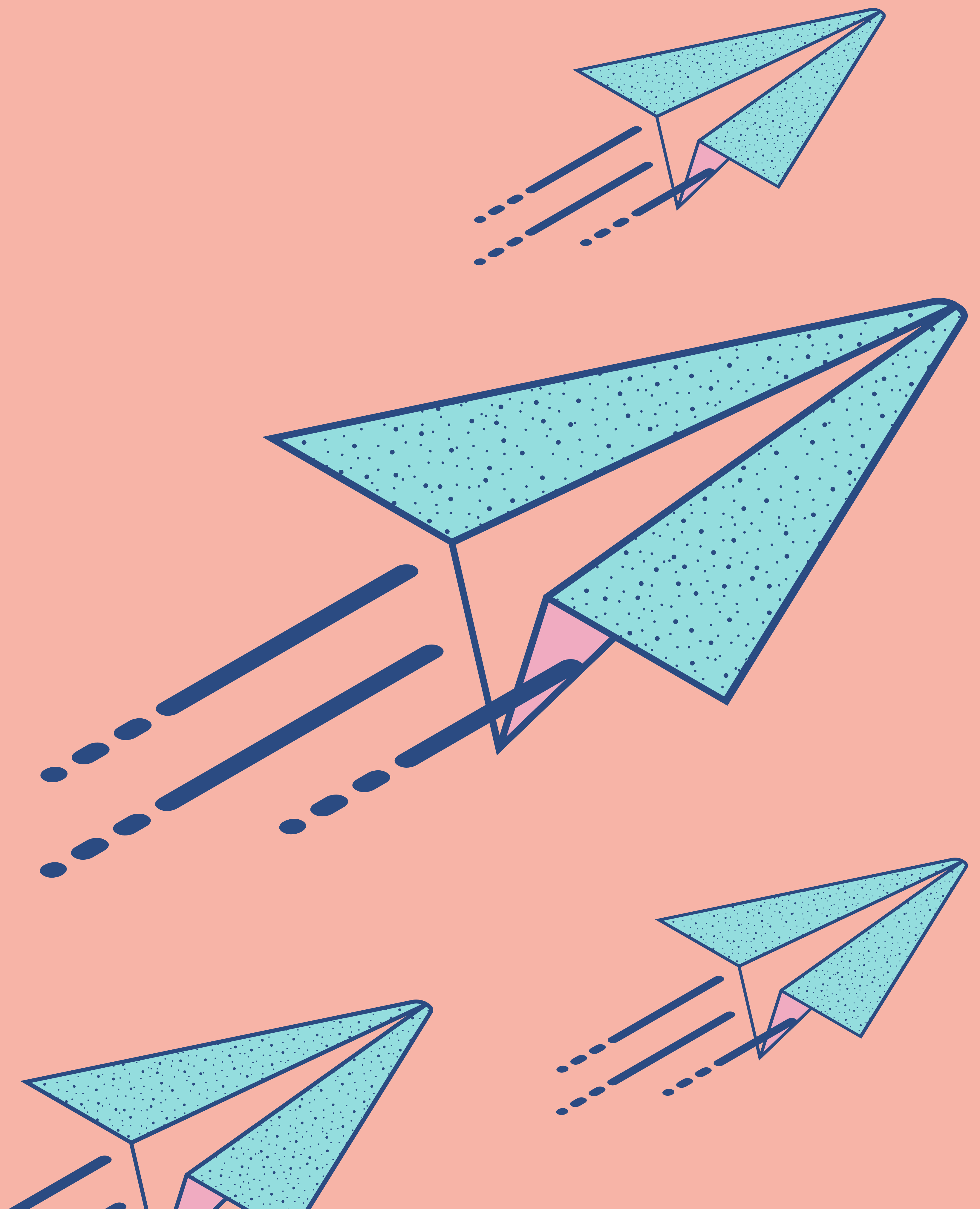
≠



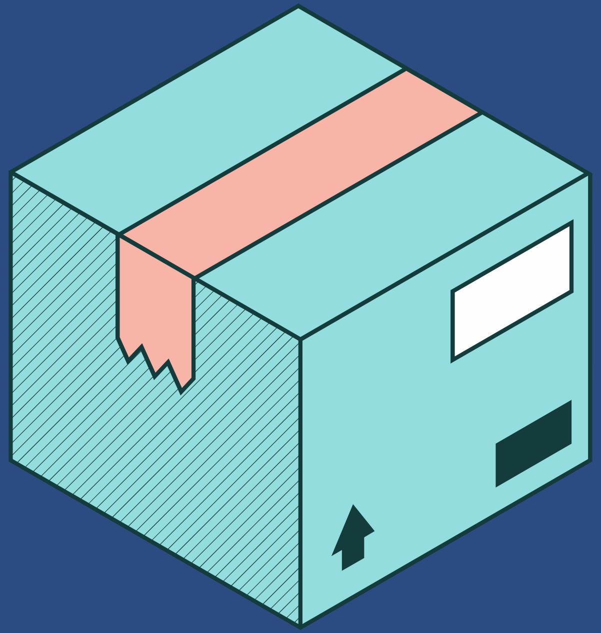
```java
public class ParallelCircuit extends Circuit{
    //Constructor
    public ParallelCircuit(){
        super();
    }

    public void setEqResistance(){
        ComplexNumber sum= new ComplexNumber(real:0, imaginary:0);
        for(ElectricComponent i: this.elements){
            i.setResistance(this.source);
            if(i.getResistance().isZero()){
                this.isShortCircuit= true;
                return;
            }
            else if(i.getResistance().isInfinity())
                sum= sum.add(new ComplexNumber(real:0, imaginary:0));

            else
                sum= sum.add(new ComplexNumber(real:1, imaginary:0).divide(i.getResistance()))
        }

        this.eqResistance= new ComplexNumber(real:1, imaginary:0).divide(sum);
    }
```

Figure 5: Inheritance & polymorphism

CONCLUSION



- Our electrical circuit simulator has successfully created a user-friendly tool for designing and analyzing electrical circuits, capable of handling both serial and parallel configurations.



- Future enhancements includes:
  a. Expanding the range of available components.
  b. Improving visualization features, and adding advanced analysis tools.

⟶ These improvements would make the simulator more powerful and versatile.

Thanks for listening